PTO/SB/17 (07-07)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no persons are required to respond to a collection of information unless it displays a valid OMB control number

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL For FY 2007

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 500

Complete if Known	
Application Number	10/716,287
Filing Date	11/18/2003
First Named Inventor	Sriram Devanathan
Examiner Name	Farhan M. Syed
Art Unit	2165
Attorney Docket No.	PQH03-037

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): ______

[✓] Deposit Account Deposit Account Number: 19-3790 Deposit Account Name: Unisys Corporation

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☐ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

[✓] Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 [✓] Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
Application Type	Fee ($)	Small Entity Fee ($)	Fee ($)	Small Entity Fee ($)	Fee ($)	Small Entity Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims ______ - 20 or HP = **Extra Claims** ______ x **Fee ($)** ______ = **Fee Paid ($)** ______

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims ______ - 3 or HP = **Extra Claims** ______ x **Fee ($)** ______ = **Fee Paid ($)** ______

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims **Fee ($)** ______ **Fee Paid ($)** ______

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets ______ - 100 = **Extra Sheets** ______ / 50 = **Number of each additional 50 or fraction thereof** ______ (round **up** to a whole number) x **Fee ($)** ______ = **Fee Paid ($)** ______

4. OTHER FEE(S)

	Fees Paid ($)
Non-English Specification, $130 fee (no small entity discount)	
Other (e.g., late filing surcharge): Appeal Brief	500

SUBMITTED BY

Signature	[illegible]	Registration No. (Attorney/Agent) 41,839	Telephone 949-380-5643
Name (Print/Type)	Phuong-Quan Hoang		Date 08/13/2007

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application. No.	:	**10/716,287**	Confirmation No. 6073
Inventor(s)	:	Sriram Devanathan *et al*	
Filed	:	November 18, 2003	
TC / Art Unit	:	2165	
Examiner	:	Syed, Farhan M.	
Docket No.	:	PQH03-037	
Customer No.	:	34225	

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Applicant submits, the following Appeal Brief pursuant to 37 C.F.R. § 41.37 for consideration by the Board of Patent Appeals and Interferences. Please charge any additional fees or credit any overpayment to our deposit Account No.19-3790. A duplicate copy of the Fee Transmittal is enclosed for this purpose.

```
08/21/2007 HVUONG1  00000100 193790    10716287
01 FC:1402         500.00 DA
```

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, Unisys Corporation.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to the appellants, the appellants' legal representative, or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1, 3-21, 23-41, 43-60 of the present application are pending. Claims 1, 3-21, 23-41, 43-60 remain rejected. Applicant hereby appeals the rejection of claims 1, 3-21, 23-41, 43-60.

IV. STATUS OF AMENDMENTS

On October 12, 2006, Applicant filed an amendment, in response to a first Office Action issued on May 12, 2006. On January 11, 2007, the Examiner issued a Final Office Action. On April 11, 2007, Applicant filed a Notice of Appeal and a Pre-Appeal Brief Request For Review in response to the Final Office Action. No amendments to the claims have been filed subsequent to the Final Office Action. On May 3, 2007, the Pre-Appeal Review Panel issued a Notice of Panel Decision from Pre-Appeal Brief Review stating that the application remains under appeal with claims 1, 3-21, 23-41, 43-60 rejected.

V. SUMMARY OF CLAIMED SUBJECT MATTER

1. Independent claims 1, 21, and 41:

In the Common Warehouse Model (CWM), the logical information or aspects are represented by entity-relationship (ER) diagrams, while the physical aspects are represented by relational elements in the relational design process where the end product is the structure of the database itself. The data to be transformed is stored in the common warehouse model (CWM) which defines a structure for the data.[1]

Physical information, also called physical aspects, of the CWM are transformed into elements that are outputted to a database management system.[2]

Items in the CWM are referred to as ER <name> for items in logical aspects, or relational <name> for items in physical aspects. Some items from the CWM that are common to both ER and relational worlds are prefixed by CWM. For the output, physical elements typically found in a DBMS or in the physical modeling of DBMS provided in database design tools are referred to as DBMS <name>.[3]

The CWM conversion system 45 reads the CWM representations 203 stored in the storage system 202 via the Application Programming Interface (API) 204 of the storage system 202, and transform the CWM representations into corresponding relational database elements. The type of data to be read from the storage system 202 and the format of the output of the CWM conversion system depend on the type of the output recipient. If the output recipient is a database management system (DBMS) 210, the CWM conversion system 45 transforms the physical aspects of the CWM to corresponding database management system (DBMS) representation in a relational database. The CWM conversion system 45 communicates with the database management system (DBMS) 210 via the API 212 of the DBMS 210.[4].

The CWM conversion system 45 processes a CWM representation in a hierarchical manner in order to transform the CWM representation into elements of a relational database. The CWM conversion system 45 processes the topmost (logical or physical) elements first. The high-level structure generated so far is outputted to the database design tool 206 or the DBMS 210. This sets up the framework for the output of the rest of the conversion of the CWM[5].

Process 1200 converts the physical aspects of a common warehouse model (CWM) to corresponding database management system (DBMS) representation in a relational database. The physical aspects of the CWM comprise relational catalogs. Each of the relational catalogs comprises relational schemas. The corresponding DBMS representation

[1] See Specification, page 6, lines 18-22; lines 27-28.
[2] See Specification, page 6, lines 10-12..
[3] See Specification, page 7, lines 19-25.
[4] See Specification, page 12, lines 4-10; 14-18; Figure 2.
[5] See Specification, page 12, lines 22-28; Figure 2.

comprises DBMS catalogs. Each of the DBMS catalogs comprises DBMS schemas. Process 1200 can output the DBMS representation to a DBMS. This output is typically in the form of a script containing instructions to modify existing or create new elements in the relational database. For example, the script used in Structured Query Language (SQL) could be the form of this output. Process 1200 scans through the relational catalogs. For each of the relational catalogs, that is being scanned, process 1200 creates a corresponding DBMS catalog to be outputted to the DBMS. For each of the relational schemas in each of the relational catalogs, process 1200 creates a corresponding DBMS schema in the corresponding DBMS catalog to hold the corresponding information. Process 1200 processes each of the relational schemas to produce the corresponding information for the corresponding DBMS schema. [6]

2. Dependent claims 3-20, 23-40, and 43-60:

Process 1200 processes each of the relational schemas in the CWM independently of the other relational schemas.[7]

Process 1300 processes a relational schema in the CWM to produce a corresponding DBMS schema for the relational database. First, process 1300 processes CWM data types included in the CWM. Process 1300 creates DBMS data types corresponding to the CWM data types. Process 1300 processes relational tables included in the relational schema. Process 1300 processes relational foreign key relationships for each of the relational tables. Process 1300 processes the checkconstraints for the relational schema. Process 1300 creates the DBMS tables corresponding to the relational tables. Process 1300 processes the relational views for the relational schema. Process 1300 processes the relational indices for the relational schema. Process 1300 processes the relational triggers for the relational schema. Process 1300 processes the relational procedures for the relational schema. Process 1300 then terminates.[8]

Process 1400 processes the data types in the CWM to produce the corresponding DBMS data types for the relational database. Process 1400 enumerates all CWM data types so that these CWM data types can be represented by the DBMS data types. Process 1400 sets pointer to the first CWM data type. Process 1400 determines whether the CWM

[6] See Specification, page 19, line 16 through page 20, line 4; Figure 12.
[7] See Specification, page 20, lines 4-5; Figure 12.
[8] See Specification, page 20, line 21 through page 21, line 4; Figure 13.

data type is a default data type, i.e., matches with a DBMS-provided data type, or a user-defined data type. A user-defined data type has a base type (which should match one of the default DBMS-provided data types) and at least one rule or constraint. If the CWM data type is user-defined, process 1400 obtains the base type and the constraint(s). A constraint is also called a rule. A data type can be dates, numbers, text, or Boolean. Process 1400 checks whether the CWM data type is text. If it is text, process 1400 obtains the character set, name of language and collation sets associated with the CWM data type. The name of language and the collation sets together define textual characteristics and sorting order. Process 1400 then increases the pointer. Process 1400 determines if there is another CWM data type from the list of data types to be processed. If there is, process 1400 proceeds to process this CWM data type as before. Otherwise, process 1400 specifies the default character set for the DBMS output then terminates. It is noted that it is usually not possible to add character sets to a DBMS. If the DBMS supported character sets are more limited than what is required, the default character set or a closely related character set will be used instead.[9]

Process 1500 creates DBMS data types corresponding to the user-defined CWM data types. Process 1500 sets pointer to the first element of the list of the user-defined CWM data types and determines whether there is a user-defined CWM data type. If there is, then process 1500 creates a corresponding DBMS data type in the corresponding DBMS schema. Process 1500 sets the underlying physical type for the DBMS data type, based on the previously obtained base type of the CWM data type and binds a constraint or constraints to the DBMS data type, based on the previously obtained constraint or constraints of the CWM data type. It is noted that, if the output recipient, i.e., the DBMS, does not provide this capability of binding a constraint to a data type, then the constraint will have to be added to every DBMS column that uses this data type.[10]

Process 1600 processes relational tables included in a relational schema. Process 1600 sets a first pointer to the first element of the list of relational tables in the relational schema being considered. Process 1600 determines whether the relational table exists. If it does not exist, process 1600 terminates. Otherwise, process 1600 sets a second pointer to the first column of this relational table. Process 1600 obtains the properties of the column. These properties include the type, precision, scale, length, IsNullable,

[9] See Specification, page 21, lines 5-29; Figure 14.

CollationName, and CharactersetName. Process 1600 verifies that the type of this column matches one of the existing DBMS data types which include the default DBMS-provided data types and the user-defined DBMS data types previously created by process 1500. Note that, if the type of this column does not match an existing DBMS data types, process 1600 would flag an error. The relational table has a relational primary key. Process 1600 determines whether this relational column is part of the relational primary key. If the relational column is part of the relational primary key, process 1600 flags this column to so indicate. Process 1600 then proceeds to process any remaining columns in the relational table.[11]

Process 1700 processes relational foreign key relationships of a relational table. Process 1700 is executed for each of the relational tables. Process 1700 enumerates all the relational tables that have foreign key relationships with the relational table being considered. The relational table being considered is referred to as the "parent" relational table, and the relational tables with which the parent relational table has foreign key relationships are referred to as "child" relational tables. While enumerating the child relational tables, for each of the child relational tables, i.e., for each of these foreign key relationships, process 1700 enumerates all the relational columns imported from the respective child relational table to the parent relational table. While enumerating the relational columns, process 1700 obtains the properties of each of the columns being enumerated, including "update" and "delete" referential integrity rules and deferability type.[12]

Process 1800 processes relational checkconstraints for a relational schema. Process 1800 is executed for each of the relational schemas. Process 1800 enumerates all relational checkconstraints associated with the relational schema being considered. While enumerating a relational checkconstraint, process 1800 obtains the parameters that describe this relational checkconstraint and enumerates all relational columns that have references to this relational checkconstraint. These are the relational columns that are operated on by this checkconstraint. The checkconstraint has pointers to the relational columns that it affects.[13]

[10] See Specification, page 22, lines 1-14; Figure 15.
[11] See Specification, page 22, line 16 through page 23, line 6; Figure 16.
[12] See Specification, page 23, lines 7-20; Figure 17.
[13] See Specification, page 24, lines 3-10, lines 17-19; Figure 18.

Process 1900 creates DBMS tables that represent the relational tables included in a relational schema. The foreign key relationships identified previously (by process 1700) represent dependencies between the relational tables in a relational schema. In general, when outputting to a script language, it is desirable to have the definition of a DBMS table preceded by the DBMS tables that this DBMS table depends on. To achieve that, process 1900 first selects the relational tables that have no dependency on any other relational tables and output these selected relational tables as DBMS tables. Next, process 1900 iterates through the remaining relational tables, selects and outputs one at a time the relational tables that depend only on at least one of the previously selected (and outputted) tables. This step is repeated until an iteration is completed without any DBMS table being created. This means that, at this point, there is no relational table left that depends only on at least one of the previously selected (and outputted) tables. If there are any remaining relational tables that have not been selected, process 1900 outputs each of these remaining relational tables using forward references. Note that, if there are any relational tables left that have not been selected, it is because they are mutually dependent. Assuming that the output DBMS language supports forward references, a DBMS table can be created for each of these mutually dependent relational tables. By "forward references" it is meant that a DBMS table in the process of being created is defined in terms of dependencies on DBMS tables that have not yet been created. Another technique for creating DBMS tables for the mutually dependent relational tables is to partially define the DBMS tables and then use ALTER TABLE commands to modify the partially defined DBMS tables. If the output DBMS language does not support forward references or ALTER TABLE commands, then the physical aspects of the CWM cannot be transformed successfully to this DBMS. [14]

Process 2000 creates a DBMS table to represent a relational table. Process 2000 creates a DBMS table having DBMS columns representing respectively the columns of the relational table. First, process 2000 creates a DBMS table. This can be done by, for example, issuing a script "create table" statement. Process 2000 sets pointer to the first column of the relational table. Process 2000 creates a DBMS column to represent this relational column. Process 2000 sets the properties, including precision, scale, length, data type, IsNullable, CollationName, and CharactersetName for the newly created DBMS column in accordance with the respective properties of the corresponding relational

[14] See Specification, page 24, line 22 through page 25, line 17; Figures 19A and 19B.

column. Process 2000 queries whether this relational column is (1) neither a primary key nor a foreign key, or (2) part of the primary key or foreign key but not the only one representing the primary key or foreign key, or (3) the only column that represents the primary key or foreign key. If it is (1) neither a primary key nor a foreign key, process 2100 proceeds to query about checkconstraint. If it is (2) part of the primary key or foreign key but not the only one representing the primary key, process 2000 flags the DBMS column to so indicate then proceeds to query about checkconstraint. If it is (3) the only column that represents the primary key or foreign key, then process 2000 specifies this property of "primary key" or "foreign key" for the DBMS column and proceeds to query about checkconstraint. Process 2000 specifies a DBMS primary key by specifying the properties of the DBMS primary key in accordance with the properties of the relational primary key as stored in the CWM. Process 2000 specifies a DBMS foreign key by specifying the "child" DBMS table and the DBMS columns of this child DBMS table that are being imported, and specifies the properties of DBMS foreign key, including the "update" and "delete" referential integrity rules and the deferability type, in accordance with the properties of the corresponding relational foreign key.

Process 2000 queries whether there is any checkconstraint associated with this relational column. The result of this query can only be one of the following: (1) there is no checkconstraint associated with this relational column, (2) there is a checkconstraint(s) associated with this relational column that does not involve any of the other relational columns, (3) there is a checkconstraint(s) associated with this relational column that also involves one or more of the other relational columns. If the query result is (1), process 2000 proceeds to work on the next column of the relational table. If the query result is (2), the process 2000 specifies the checkconstraint(s) at column-level in the DBMS column then proceeds to work on the next column of the relational table. Process 2000 specifies a checkconstraint by specifying the parameters, i.e., the constraint expressions that describe the checkconstraint. If the query result is (3), process 2000 flags the DBMS column to so indicate then proceeds to work on the next column of the relational table. Process 2000 increases the pointer to the next column of the relational table. Process 2000 checks whether the column exists. If the column exists, process 2000 proceeds to work on it as previously described. If the relational column does not exist, this means that all the necessary DBMS columns have been created. Process 2000 checks whether there is any checkconstraint(s) that involves multiple columns of the relational table, which consequently, involves multiple columns of the corresponding DBMS table (block 2022). If there is, process 2000 specifies the checkconstraint(s) at the table level in the DBMS table and identifies the DBMS columns that are associated with this checkconstraint, then proceeds to block 2026. If there is no such checkconstraint(s), process 2000 checks whether there is a multi-column primary key or a multi-column foreign key in the relational table. If there is none, process 2000 terminates. Otherwise, process 2000 specifies the multi-column primary key or the multi-column foreign key in the DBMS table at table level and identifies the DBMS columns that represent the multi-column primary key or the multi-column foreign key.[15]

Process 2100 processes the relational views for a relational schema. Process 2100 is executed for each of the relational schemas. Process 2100 sets a pointer to the first element in the list of relational views. Process 2100 checks whether this relational view exists. If it does not exist, process 2100 terminates. If it exists, process 2100 creates a DBMS view to represent this relational view. A relational view has two properties, namely, query expression and updatability. Query expression defines the relational view and is the main property of the relational view. Updatability is based on the IsReadOnly property. Process 2100 reads the query expression of the relational view and specifies the query expression for the DBMS view accordingly. Process 2100 reads the updatability of the relational view and specifies the updatability for the DBMS view accordingly. Process 2100 increases the pointer to point to the next relational view, checks whether this next relational view exists and proceeds as previously described.[16]

Process 2200 processes the relational indices for a relational schema. Process 2200 is executed for each of the relational schemas. Process 2200 sets a pointer to the first element in the list of relational indices. Process 2200 checks whether this relational index exists. If it does not exist, process 2200 terminates. If it exists, process 2200 creates a DBMS index to represent this relational index. This relational index uses certain relational columns. Process 2200 specifies all the DBMS columns that correspond to these relational columns. Process 2200 sets the parameters for the DBMS index, including IsNullable, FilterCondition, and AutoUpdate parameters, in accordance with the parameters of the

[15] See Specification, page 26, line 24 through page 28, line 20; Figure 20.
[16] See Specification, page 28, line 21 through page 29, line 5; Figure 21.

relational index. Process 2200 increases the pointer to point to the next relational index, checks whether this next relational index exists then proceeds as previously described.[17]

Process 2300 processes the relational triggers for a relational schema. Process 2300 is executed for each of the relational schemas. Process 2300 sets a pointer to the first element in the list of relational triggers. Process 2300 checks whether this relational trigger exists. If it does not exist, process 2300 terminates. If it exists, process 2300 creates a DBMS trigger to represent this relational trigger. The relational trigger's properties are described by parameters. The parameters specify when the relational trigger is invoked and what action is to be taken when it is invoked. Process 2300 sets the parameters for the DBMS trigger that corresponds to the relational trigger in accordance with the relational trigger's parameters. The relational trigger monitors a specific relational table that corresponds to a specific DBMS table. Process 2300 sets this specific DBMS table for the DBMS trigger, i.e., provides the DBMS trigger with a link to this DBMS table. Process 2300 increases the pointer to point to the next relational trigger, checks whether this next relational trigger exists and proceeds as previously described.[18]

Process 2400 processes the relational procedures for a relational schema. Process 2400 is executed for each of the relational schemas. Process 2400 sets a pointer to the first element in the list of relational procedures. Process 2400 checks whether this relational procedure exists. If it does not exist, process 2400 terminates. If it exists, process 2400 creates a DBMS procedure to represent this relational procedure. The relational procedure is described by a set of arguments. Process 2400 sets the arguments for the DBMS procedure in accordance with the arguments of relational procedure. Process 2400 increases the pointer to point to the next relational procedure, checks whether this next relational procedure exists then proceeds as previously described.[19]

[17] See Specification, page 29, line 6 through page 29, line 20; Figure 22.
[18] See Specification, page 29, line 21 through page 30, line 7; Figure 23.
[19] See Specification, page 30, lines 8-20; Figure 24.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 21, 23-40 stand rejected under 35 U.S.C. §101 as being unpatentable for being directed to non-statutory subject matter.

Claims 1, 3-21, 23-41, 43-60 stand rejected under 35 U.S.C. §102(b), as being anticipated by a non-patent literature titled "Designing and Creating Relational Schemas with a CWM-Based Tool" by Kumpon Farpinyo *et al*, pages 456-461, 2002, Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand (hereinafter, "Farpinyo")..

VII. ARGUMENTS

A. Claims 21, 23-40 Are Not Unpatentable under 35 U.S.C. §101.

In the Final Office Action, the Examiner maintained his rejection of claims 21, 23-40 under 35 U.S.C. §101, stating "The Examiner disagrees with the Applicant's analysis of a carrier wave. The Examiner is not refuting the use of RF waves, but instead the use of the words carrier wave or a signal modulated by a carrier. The Examiner will continue to rely on the Interim Guidelines, as explained in the previous office action dated 12 May 2006, as the basis for rejection of claims 21-40" (Final Office Action, pages 3-4, item 6).

In response to the first Office action, Applicant has amended claim 21 to limit claims 21 and its dependent claims to machine-accessible storage medium in order to obtain a timely Notice of Allowance.

The Examiner repeated the rejection without taking note of the Applicant's amendment as presented in the previously filed response. The MPEP requires that the Examiner's action will be complete as to all matters. 37 CRF 1.104; MPEP 707.07. Since the Examiner's action in the Office Action is incomplete in that there is no answer to the substance of Applicant's amendment previously presented, the rejections have been improperly made.

B. Claims 1, 3-21, 23-41, 43-60 Are Not Anticipated by Farpinyo.

In the Final Office Action, the Examiner rejected claims 1, 3-21, 23-41, 43-60 under 35 U.S.C. §102(b), as being anticipated by a non-patent literature titled "Designing and Creating Relational Schemas with a CWM-Based Tool" by Kumpon Farpinyo *et al*, pages 456-461, 2002, Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand (hereinafter, "Farpinyo"). Applicant respectfully traverses the rejection and submits that the Examiner has not met the burden of establishing a prima facie case of anticipation.

Farpinyo discloses that a tool called ER2CWM can create CWM relational database schemas from physical data models represented by ER diagrams (Abstract). The tool supports the creation of ER diagrams through its ER editor, generates CWM Relational metadata, and creates database schemas. It can also transform database schemas back into CWM Relational metadata and ER diagrams respectively (Farpinyo, page 457, last 3 lines, page 458, first 2 lines).

Farpinyo discloses that the tool ER2CWM considers only the part of CWM for relational database schemas called CWM Relational (Farpinyo, Section 1, page 456, last 2 lines). As well known in the art, CWM Relational is the part that contains the physical information. Refer, for example, to this OMG weblink:

http://www.omg.org/docs/formal/03-03-29.pdf

In Section 2 (Farpinyo, page 458), Farpinyo gave a brief introduction to CWM Relational by explaining several elements of CWM Relational, such as Catalog, Schema, Table, Column, InitialValue, CheckConstraint, PrimaryKey, ForeignKey, SQL data type.

In Section 3 (Farpinyo, page 459), Farpinyo discloses that the tool ER2CWM is composed of four modules: ER Editor, Metadata, DBMS Information, and ER Module. The ER Editor is the editor for designing physical data models with ER diagrams. The ER Editor is a GUI for user to create CWM Relational metadata, select DBMSs to create database schemas, or create CWM Relational metadata and ER diagrams from existing relational databases. The Metadata module creates and maintains two types of metadata: diagram metadata and CWM metadata. Diagram metadata (DIA) is the metadata of ER models with display information for the diagrams. CWM metadata represents ER models or database schemas and conforms to CWM v1.0. The DBMS Information module creates database schemas from CWM Relational metadata, reads in existing database schemas to create CWM Relational metadata and ER diagrams, and maintains information about DBMSs that the tool supports. The ER Module connects together the other three modules. It contacts DBMS Information module when database designers select DBMSs to design physical data models or to create database schemas. It interacts with the Metadata module to get and save DIA and CWM Relational metadata. (Farpinyo, Section 3, page 459).

Note that, although Farpinyo uses the term "CWM metadata" in the description of the Metadata module, Farpinyo states that the ER Module interacts with the Metadata module to get and save DIA and CWM Relational metadata. Thus, what is called CWM metadata in the description of the Metadata module is actually CWM Relational metadata. Farpinyo further clarifies that the tool "ER2CWM cannot properly display ER diagrams that are not created by the tool itself. That is, ER diagrams that are created from any CWM documents or created from pre-existing database schemas do not have associated .DIA files that ER2CWM needs for a proper display. Database designers will have to arrange the layout of the diagrams by themselves." (Farpinyo, page 460, Footnote 1).

In Section 4 (Farpinyo, page 460), an example is given with an ER diagram, CWM metadata, and generated database schema as results of using the tool. A database designer first selects Sybase Adaptive Server (a DBMS) as a target of the design, and draws an ER diagram on the GUI of the tool. The tool then generates a corresponding CWM Relational metadata for this design. The designer later changes to create a database schema for MS SQL Server (another DBMS) instead by using the CWM metadata generated earlier.

Farpinyo discloses that a user can specify a target DBMS and can input an ER diagram to the ER2CWM tool by drawing it using the GUI of the tool. The ER2CWM tool then uses the ER diagram to generate and store a corresponding CWM Relational metadata. When the user later specifies a different DBMS, the ER2CWM tool uses the stored CWM Relational metadata to generate a database schema for this different DBMS (Farpinyo, page 460, Section 4).

Farpinyo discloses a user manual of the ER2CWM tool to show how to use the GUI of the ER2CWM tool to create an ER diagram (Farpinyo, User Manual, page 2 through page 7, first paragraph); to create a database schema from the ER diagram Farpinyo, User Manual, page 7); to read a schema from a specified database (Farpinyo, User Manual, pages 8-9); to create an ER diagram from a selected CWM Metadata file (Farpinyo, User Manual, pages 9-10); to save an ER diagram into HTML format (Farpinyo, User Manual, page 11, item 5); and to print an ER diagram (Farpinyo, User Manual, page 11, item 6).

<u>Farpinyo</u> merely discloses that the ER2CWM tool can receive an ER diagram as input from a user, generate and store corresponding CWM Relational metadata, and generate a database schema for a specified DBMS using the stored CWM Relational metadata. <u>Farpinyo</u> does not disclose the methodology of converting the generated CWM Relational metadata to a database schema for a specified DBMS.

<u>Farpinyo</u> does not disclose, either inherently or explicitly, at least one of the following elements: (1) converting physical aspects of a common warehouse model (CWM) to corresponding database management system (DBMS) items in a relational database by processing in a hierarchical manner the physical aspects and creating the corresponding DBMS items, the physical aspects comprising relational catalogs, the relational catalogs comprising relational schemas, the corresponding DBMS items comprising DBMS catalogs, the DBMS catalogs comprising DBMS schemas, wherein converting comprises the operations of: (a) scanning through the relational catalogs; (b) for a first of the relational catalogs, creating a corresponding first DBMS catalog in the relational database; (c) for each of the relational schemas in the first relational catalog, creating a corresponding DBMS schema in the corresponding DBMS catalog to hold corresponding information; and (d) processing each of the relational schemas to produce corresponding information for the corresponding DBMS schema.

To anticipate a claim, the reference must teach every element of the claim. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." Vergegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 631, 2 USPQ 2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the...claim." Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 USPQ 2d 1913, 1920 (Fed. Cir.1989). Since the Examiner failed to show that <u>Farpinyo</u> teaches or discloses any of the elements of the claims, the rejection under 35 U.S.C. §102 is improper.

In response to Applicant's argument that <u>Farpinyo</u> does not disclose the methodology of converting the generated CWM Relational metadata to a database schema for a specified DBMS, the Examiner stated that "the methodology of the use of ER2CWM is clearly anticipated by prior art of record, wherein Sections 3 and 4 clearly illustrate the use of ER2CWM. An ordinary person skilled in the art clearly understands that the steps required to execute a tool like ER2CWM to create CWM relational database schemas from physical data models represented by ER diagrams must include the steps that are recited in claims 1, 21, 41. This methodology is fundamental, let alone the essence of creating relational schemas with a CWM-based tool." (Final Office Action, page 4, item 7).

Applicant respectfully disagrees for the following reasons.

First, the Examiner stated that "the methodology of the use of ER2CWM is clearly anticipated by prior art of record", but did not cite what prior art of record the Examiner had in mind.

Second, it appears that the Examiner applied the theory of inherency or relied on official notice to arrive at the conclusion that Farpinyo anticipates claims 1, 3-21, 23-41, 43-60. Applicant submits that the Examiner's reliance of the theory of inherence or official notice is misplaced for the following reasons.

First, the fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic. In re Rijckaert, 9 F.3d 1531, 1534, 28 USPQ2d 1955, 1957 (Fed. Cir. 193). "To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill. Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.'" In re Robertson, 169 F.3d 743, 745, 49 USPQ2d 1949, 1950-51 (Fed. Cir. 1999). "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art." Ex parte Levy, 17 USPQ2d 1461, 1464 (Bd. Pat. App. & Inter. 1990) (emphasis in original). Here, Farpinyo does not disclose any of the claimed operations involved in converting physical aspects of CWM to DBMS elements. The Examiner has not shown that such CWM conversion operations necessarily flow from the teachings of Farpinyo.

Second, official notice unsupported by documenting evidence should only be taken by the Examiner where the facts asserted to be well-known, or to be common knowledge in the art are capable of instant and unquestionable demonstration as being well known. In re Ahlert, 424 F.2d 1088, 1091, 165 USPQ 418, 420 (CCPA 1970); MPEP 2144.03A. It would not be appropriate for the Examiner to take official notice of facts without citing a prior art reference. MPEP 2144.03A. Furthermore, if official notice is taken of a fact, unsupported by documentary evidence, the technical line of reasoning underlying a decision to take such notice must be clear and unmistakable. MPEP 2144.03B. Here, Farpinyo neither discloses nor suggests **how to convert** CWM Relational metadata to database schema. Farpinyo merely states that the tool is capable of doing this for certain DBMSs (Farpinyo, page 461, Section 5). The Examiner did not provide a technical line of reasoning which must be clear and unmistakable. The Examiner merely states that "an ordinary person skilled in the art clearly understands that the steps required to execute a tool like ER2CWM to create CWM relational database schemas from physical data models represented by ER diagrams must include the steps that are recited in claims 1, 21, 41. This methodology is fundamental, let alone the essence of creating relational schemas with a CWM-based tool." (Final Office Action, page 4, item 7). Therefore, the Examiner's reasoning is not clear and not unmistakable.

Furthermore, it appears that the Examiner did not appreciate that the physical aspects of CWM and the logical aspects of CWM are two different levels of information in CWM. The Examiner's confusion is apparent throughout item 9 of the Final Office Action (Final Office Action, item 9, pages 5-20). For example, in contending that Farpinyo anticipates the operation of "scanning through the relational catalogs", the Examiner stated that "it is clear that in order to create and read schemas, scanning of ER libraries must be performed. Figure 3 clearly illustrates such example" (Final Office Action, page 6, lines 21-22). ER libraries are elements of the logical aspects of CWM, while relational catalogs are elements of the physical aspects of CWM. The claims in the present application do not address any element that exist only in the logical aspects of CWM. A further example: in contending that Farpinyo anticipates the element "each of the relational schemas is processed independently", the Examiner stated that "Figures 1-11 steps through the process of creating an ER model and then converting it into a relational database. This process is a continuous process, where each ER model is created independently with each other, until the user completes the desired relational database specifications." (Final Office Action, page 8, second full paragraph). ER models are elements of the logical aspects of CWM, while relational schemas are elements of the physical aspects of CWM. The claims in the present application do not address any element that exists only in the logical aspects of CWM.

Therefore, Applicant submits that claims 1, 3-21, 23-41, 43-60 are distinguishable over the cited prior art reference.

VIII. CONCLUSION

Applicant respectfully requests that the Board enter a decision overturning the Examiner's rejection of all pending claims, and holding that the claims satisfy the requirements for patentability of 35 U.S.C. §101, and 35 U.S.C. §102(b).

Respectfully submitted,

UNISYS CORPORATION

Dated: August 13, 2007 By Phuong-Quan Hoang

Phuong-Quan Hoang
Reg. No. 41,839
Tel.: 949-380-5643 (Pacific Coast)
Fax: 949-380-5254

Unisys Corporation
25725 Jeronimo Road, MS 400
Mission Viejo, CA 92691

IX. CLAIM APPENDIX

The claims of the present application which are involved in this appeal are as follows:

1. (previously presented) A method comprising:

converting physical aspects of a common warehouse model (CWM) to corresponding database management system (DBMS) items in a relational database by processing in a hierarchical manner the physical aspects and creating the corresponding DBMS items, the physical aspects comprising relational catalogs, the relational catalogs comprising relational schemas, the corresponding DBMS items comprising DBMS catalogs, the DBMS catalogs comprising DBMS schemas, wherein converting comprises the operations of:

(a) scanning through the relational catalogs;

(b) for a first of the relational catalogs, creating a corresponding first DBMS catalog in the relational database;

(c) for each of the relational schemas in the first relational catalog, creating a corresponding DBMS schema in the corresponding DBMS catalog to hold corresponding information; and

(d) processing each of the relational schemas to produce corresponding information for the corresponding DBMS schema.

2. (canceled)

3. (previously presented) The method of Claim 1 wherein, in operation (d), each of the relational schemas is processed independently.

4. (original) The method of Claim 1 wherein operation (d) comprises:

(1) processing CWM data types included in a first of the relational schemas;

(2) creating DBMS data types corresponding to the CWM data types;

(3) processing relational tables included in the first relational schema;

(4) processing relational foreign key relationships for each of the relational tables;

(5) processing relational checkconstraints for the first relational schema;

(6) creating DBMS tables corresponding to the relational tables;

(7) processing relational views for the first relational schema;

(8) processing relational indices for the first relational schema;

(9) processing relational triggers for the first relational schema; and

(10) processing relational procedures for the first relational schema.

5. (original) The method of Claim 4 wherein (1) processing CWM data types included in a first of the relational schemas comprises:

for one of the CWM data types, determining whether the CWM data type is user-defined;

if the CWM data type is user-defined, obtaining base type and constraint of the CWM data type; and

if the CWM data type is text, obtaining a character set, name of language and collation sets associated with the CWM data type.

6. (original) The method of Claim 5 wherein (2) creating DBMS data types corresponding to the CWM data types comprises:

for a first of the CWM data types that is user-defined,

creating a corresponding DBMS data type in the corresponding DBMS schema;

setting physical type for the DBMS data type, based on the obtained base type of the first CWM data type; and binding a constraint to the DBMS data type, based on the obtained constraint of the first CWM data type.

7. (original) The method of Claim 6 wherein (3) processing relational tables included in the first relational schema comprises:

determining whether there is a first relational table in the first relational schema;

if there is a first relational table in the first relational schema, then:

determining relational columns in the first relational table, the first relational table having a relational primary key; and, for each of the relational columns:

obtaining column properties including type, precision, scale, length, IsNullable, CollationName, and CharactersetName;

verifying that the obtained type matches one of the DBMS data types;

determining whether the relational column is part of the relational primary key; and

flagging the relational column if the relational column is part of the relational primary key.

8. (original) The method of Claim 4 wherein (4) processing relational foreign key relationships for each of the relational tables comprises:

for a first of the relational tables, enumerating child relational tables having foreign key relationships with the first relational table;

for each of the foreign key relationships,

determining relational columns imported from the respective child relational table to the first relational table; and

obtaining properties of each of the imported relational columns, including "update" and "delete" referential integrity rules and deferability type.

9. (original) The method of Claim 4 wherein (5) processing relational checkconstraints for the first relational schema comprises:

determining relational checkconstraints associated with the first relational schema;

obtaining parameters associated with a first of the relational checkconstraints; and

enumerating relational columns having references to the first relational checkconstraint.

10. (original) The method of Claim 4 wherein (6) creating DBMS tables corresponding to the relational tables comprises:

selecting from the relational tables included in the first relational schema first tables having no dependencies on any other of the relational tables; and

creating a corresponding DBMS table for each of the first selected tables.

11. (original) The method of Claim 10 further comprising:

selecting from the relational tables included in the first relational schema a second table having dependency on at least one of the first selected tables; and

creating a corresponding DBMS table for the second selected table.

12. (original) The method of Claim 11 further comprising:

selecting from the relational tables included in the first relational schema a third table having dependency on at least one of the second and the first selected tables; and

creating a corresponding DBMS table for the third selected table.

13. (original) The method of Claim 10 further comprising:

creating a corresponding DBMS table for each of mutually dependent tables from the relational tables using forward references or ALTER TABLE commands.

14. (original) The method of Claim 10 wherein creating a corresponding DBMS table comprises:

creating DBMS columns corresponding to columns of the corresponding relational table;

setting properties including precision, scale, length, data type, IsNullable, CollationName, and CharactersetName for each of the DBMS columns based on respective properties of the corresponding relational column;

if one of the DBMS columns is the only one of the DBMS columns that represents a primary key or a foreign key, adding property of primary key or foreign key to the one DBMS column; and

if there is a checkconstraint associated with one of the DBMS columns and not involving any of the remaining DBMS columns, specifying the checkconstraint as column-level constraint.

15. (original) The method of Claim 14 further comprising:

if there is a multi-column primary key or a multi-column foreign key in the relational table, specifying the multi-column primary key or a multi-column foreign key in the DBMS table at table-level and identifying the DBMS columns that represent the multi-column primary key or a multi-column foreign key; and

if there is a checkconstraint involving multiple DBMS columns, specifying the constraint in the DBMS table at table-level and identifying the involved DBMS columns.

16. (original) The method of Claim 14 further comprising:

specifying a foreign key in the DBMS table, including:

identifying a child DBMS table and DBMS columns being imported from the child DBMS table; and

specifying properties of the foreign key, the properties including "update" and "delete" referential integrity rules and deferability type.

17. (original) The method of Claim 4 wherein (7) processing relational views for the first relational schema comprises:

determining relational views associated with the first relational schema;

for each of the relational views:

creating a corresponding DBMS view;

specifying updatability of the corresponding DBMS view; and

specifying query expression defining the corresponding DBMS view.

18. (original) The method of Claim 4 wherein (8) processing relational indices for the first relational schema comprises:

determining relational indices associated with a first of the relational schemas;

for each of the relational indices:

creating a corresponding DBMS index to represent the relational index;

specifying DBMS columns used by the corresponding DBMS index; and

setting properties of the specified DBMS columns including IsNullable, FilterCondition, and AutoUpdate.

19. (original) The method of Claim 4 wherein (9) processing relational triggers for the first relational schema comprises:

determining relational triggers associated with the first relational schema;

for each of the relational triggers:

creating a corresponding DBMS trigger;

setting properties of the corresponding DBMS trigger based on properties of the relational trigger, the relational trigger monitoring a relational table; and

setting a monitored DBMS table corresponding to the monitored relational table.

20. (original) The method of Claim 4 wherein (10) processing relational procedures for the first relational schema comprises:

determining relational procedures associated with the first relational schema;

for each of the relational procedures:

creating a corresponding DBMS procedure; and

setting arguments for the corresponding DBMS procedure based on arguments of the relational procedure.

21. (previously presented) An article of manufacture comprising:

a machine-accessible medium including data that, when accessed by a machine, cause the machine to perform the operation of:

converting physical aspects of a common warehouse model (CWM) to corresponding database management system (DBMS) items in a relational database by processing in a hierarchical manner the physical aspects and creating the corresponding DBMS items, the physical aspects comprising relational catalogs, the relational catalogs comprising relational schemas, the corresponding DBMS items comprising DBMS catalogs, the DBMS catalogs comprising DBMS schemas, wherein the operation of converting comprises the operations of:

(a) scanning through the relational catalogs;

(b) for a first of the relational catalogs, creating a corresponding first DBMS catalog in the relational database;

(c) for each of the relational schemas in the first relational catalog, creating a corresponding DBMS schema in the corresponding DBMS catalog to hold corresponding information; and

(d) processing each of the relational schemas to produce corresponding information for the corresponding DBMS schema.

22. (canceled)

23. (previously presented) The article of manufacture of Claim 21 wherein, in operation (d), each of the relational schemas is processed independently.

24. (original) The article of manufacture of Claim 21 wherein operation (d) comprises:

(1) processing CWM data types included in a first of the relational schemas;

(2) creating DBMS data types corresponding to the CWM data types;

(3) processing relational tables included in the first relational schema;

(4) processing relational foreign key relationships for each of the relational tables;

(5) processing relational checkconstraints for the first relational schema;

(6) creating DBMS tables corresponding to the relational tables;

(7) processing relational views for the first relational schema;

(8) processing relational indices for the first relational schema;

(9) processing relational triggers for the first relational schema; and

(10) processing relational procedures for the first relational schema.

25. (original) The article of manufacture of Claim 24 wherein the operation of (1) processing CWM data types included in a first of the relational schemas comprises:

for one of the CWM data types, determining whether the CWM data type is user-defined;

if the CWM data type is user-defined, obtaining base type and constraint of the CWM data type; and

if the CWM data type is text, obtaining a character set, name of language and collation sets associated with the CWM data type.

26. (original) The article of manufacture of Claim 25 wherein the operation of (2) creating DBMS data types corresponding to the CWM data types comprises:

for a first of the CWM data types that is user-defined,

creating a corresponding DBMS data type in the corresponding DBMS schema;

setting physical type for the DBMS data type, based on the obtained base type of the first CWM data type; and

binding a constraint to the DBMS data type, based on the obtained constraint of the first CWM data type.

27. (original) The article of manufacture of Claim 26 wherein the operation of (3) processing relational tables included in the first relational schema comprises:

determining whether there is a first relational table in the first relational schema;

if there is a first relational table in the first relational schema, then:

determining relational columns in the first relational table, the first relational table having a relational primary key; and, for each of the relational columns:

obtaining column properties including type, precision, scale, length, IsNullable, CollationName, and CharactersetName;

verifying that the obtained type matches one of the DBMS data types;

determining whether the relational column is part of the relational primary key; and

flagging the relational column if the relational column is part of the relational primary key.

28. (original) The article of manufacture of Claim 24 wherein the operation of (4) processing relational foreign key relationships for each of the relational tables comprises:

for a first of the relational tables, enumerating child relational tables having foreign key relationships with the first relational table;

for each of the foreign key relationships,

determining relational columns imported from the respective child relational table to the first relational table; and

obtaining properties of each of the imported relational columns, including "update" and "delete" referential integrity rules and deferability type.

29. (original) The article of manufacture of Claim 24 wherein the operation of (5) processing relational checkconstraints for the first relational schema comprises:

determining relational checkconstraints associated with the first relational schema;

obtaining parameters associated with a first of the relational checkconstraints; and

enumerating relational columns having references to the first relational checkconstraint.

30. (original) The article of manufacture of Claim 24 wherein the operation of (6) creating DBMS tables corresponding to the relational tables comprises:

selecting from the relational tables included in the first relational schema first tables having no dependencies on any other of the relational tables; and

creating a corresponding DBMS table for each of the first selected tables.

31. (original) The article of manufacture of Claim 30 wherein operation (6) further comprises:

selecting from the relational tables included in the first relational schema a second table having dependency on at least one of the first selected tables; and

creating a corresponding DBMS table for the second selected table.

32. (original) The article of manufacture of Claim 31 wherein operation (6) further comprises:

selecting from the relational tables included in the first relational schema a third table having dependency on at least one of the second and the first selected tables; and

creating a corresponding DBMS table for the third selected table.

33. (original) The article of manufacture of Claim 30 wherein operation (6) further comprises:

creating a corresponding DBMS table for each of mutually dependent tables from the relational tables using forward references or ALTER TABLE commands.

34. (original) The article of manufacture of Claim 30 wherein the operation of creating a corresponding DBMS table comprises:

creating DBMS columns corresponding to columns of the corresponding relational table;

setting properties including precision, scale, length, data type, IsNullable, CollationName, and CharactersetName for each of the DBMS columns based on respective properties of the corresponding relational column;

if one of the DBMS columns is the only one of the DBMS columns that represents a primary key or a foreign key, adding property of primary key or foreign key to the one DBMS column; and

if there is a checkconstraint associated with one of the DBMS columns and not involving any of the remaining DBMS columns, specifying the checkconstraint as column-level constraint.

35. (original) The article of manufacture of Claim 34 the operation of creating a corresponding DBMS table further comprises:

if there is a multi-column primary key or a multi-column foreign key in the relational table, specifying the multi-column primary key or a multi-column foreign key in the DBMS table at table-level and identifying the DBMS columns that represent the multi-column primary key or a multi-column foreign key; and if there is a checkconstraint involving multiple DBMS columns, specifying the constraint in the DBMS table at table-level and identifying the involved DBMS columns.

36. (original) The article of manufacture of Claim 34 the operation of creating a corresponding DBMS table further comprises:

specifying a foreign key in the DBMS table, including:

identifying a child DBMS table and DBMS columns being imported from the child DBMS table; and

specifying properties of the foreign key, the properties including "update" and "delete" referential integrity rules and deferability type.

37. (original) The article of manufacture of Claim 24 wherein the operation of (7) processing relational views for the first relational schema comprises:

determining relational views associated with the first relational schema;

for each of the relational views:

creating a corresponding DBMS view;

specifying updatability of the corresponding DBMS view; and

specifying query expression defining the corresponding DBMS view.

38. (original) The article of manufacture of Claim 24 wherein the operation of (8) processing relational indices for the first relational schema comprises:

determining relational indices associated with a first of the relational schemas;

for each of the relational indices:

creating a corresponding DBMS index to represent the relational index;

specifying DBMS columns used by the corresponding DBMS index; and

setting properties of the specified DBMS columns including IsNullable, FilterCondition, and AutoUpdate.

39. (original) The article of manufacture of Claim 24 wherein the operation of (9) processing relational triggers for the first relational schema comprises:

determining relational triggers associated with the first relational schema;

for each of the relational triggers:

creating a corresponding DBMS trigger;

setting properties of the corresponding DBMS trigger based on properties of the relational trigger, the relational trigger monitoring a relational table; and

setting a monitored DBMS table corresponding to the monitored relational table.

40. (original) The article of manufacture of Claim 24 wherein the operation of (10) processing relational procedures for the first relational schema comprises:

determining relational procedures associated with the first relational schema;

for each of the relational procedures:

creating a corresponding DBMS procedure; and

setting arguments for the corresponding DBMS procedure based on arguments of the relational procedure.

41. (previously presented) A system comprising:

a processor; and

a memory coupled to the processor, the memory containing program code that, when executed by the processor, causes the processor to perform the operation of:

converting physical aspects of a common warehouse model (CWM) to corresponding database management system (DBMS) items in a relational database by processing in a hierarchical manner the physical aspects and creating the corresponding DBMS items, the physical aspects comprising relational catalogs, the relational catalogs comprising relational schemas, the corresponding DBMS items comprising DBMS catalogs, the DBMS catalogs comprising DBMS schemas, wherein the operation of converting comprises the operations of:

(a) scanning through the relational catalogs;

(b) for a first of the relational catalogs, creating a corresponding first DBMS catalog in the relational database;

(c) for each of the relational schemas in the first relational catalog, creating a corresponding DBMS schema in the corresponding DBMS catalog to hold corresponding information; and

(d) processing each of the relational schemas to produce corresponding information for the corresponding DBMS schema.

42. (canceled)

43. (previously presented) The system of Claim 41 wherein, in operation (d), each of the relational schemas is processed independently.

44. (original) The system of Claim 41 wherein operation (d) comprises:

(1) processing CWM data types included in a first of the relational schemas;

(2) creating DBMS data types corresponding to the CWM data types;

(3) processing relational tables included in the first relational schema;

(4) processing relational foreign key relationships for each of the relational tables;

(5.) processing relational checkconstraints for the first relational schema;

(6) creating DBMS tables corresponding to the relational tables;

(7) processing relational views for the first relational schema;

(8) processing relational indices for the first relational schema;

(9) processing relational triggers for the first relational schema; and

(10) processing relational procedures for the first relational schema.

45. (original) The system of Claim 44 wherein the operation of (1) processing CWM data types included in a first of the relational schemas comprises:

for one of the CWM data types, determining whether the CWM data type is user-defined;

if the CWM data type is user-defined, obtaining base type and constraint of the CWM data type; and

if the CWM data type is text, obtaining a character set, name of language and collation sets associated with the CWM data type.

46. (original) The system of Claim 45 wherein the operation of (2) creating DBMS data types corresponding to the CWM data types comprises:

for a first of the CWM data types that is user-defined,

creating a corresponding DBMS data type in the corresponding DBMS schema;

setting physical type for the DBMS data type, based on the obtained base type of the first CWM data type; and

binding a constraint to the DBMS data type, based on the obtained constraint of the first CWM data type.

47. (original) The system of Claim 46 wherein the operation of (3) processing relational tables included in the first relational schema comprises:

determining whether there is a first relational table in the first relational schema;

if there is a first relational table in the first relational schema, then:

determining relational columns in the first relational table, the first relational table having a relational primary key; and, for each of the relational columns:

obtaining column properties including type, precision, scale, length, IsNullable, CollationName, and CharactersetName;

verifying that the obtained type matches one of the DBMS data types;

determining whether the relational column is part of the relational primary key; and

flagging the relational column if the relational column is part of the relational primary key.

48. (original) The system of Claim 44 wherein the operation of (4) processing relational foreign key relationships for each of the relational tables comprises:

for a first of the relational tables, enumerating child relational tables having foreign key relationships with the first relational table;

for each of the foreign key relationships,

determining relational columns imported from the respective child relational table to the first relational table; and

obtaining properties of each of the imported relational columns, including "update" and "delete" referential integrity rules and deferability type.

49. (original) The system of Claim 44 wherein the operation of (5) processing relational checkconstraints for the first relational schema comprises:

determining relational checkconstraints associated with the first relational schema;

obtaining parameters associated with a first of the relational checkconstraints; and

enumerating relational columns having references to the first relational checkconstraint.

50. (original) The system of Claim 49 wherein the operation of (6) creating DBMS tables corresponding to the relational tables comprises:

selecting from the relational tables included in the first relational schema first tables having no dependencies on any other of the relational tables; and

creating a corresponding DBMS table for each of the first selected tables.

51. (original) The system of Claim 49 wherein operation (6) further comprises:

selecting from the relational tables included in the first relational schema a second table having dependency on at least one of the first selected tables; and

creating a corresponding DBMS table for the second selected table.

52. (original) The system of Claim 51 wherein operation (6) further comprises:

selecting from the relational tables included in the first relational schema a third table having dependency on at least one of the second and the first selected tables; and

creating a corresponding DBMS table for the third selected table.

53. (original) The system of Claim 50 wherein operation (6) further comprises:

creating a corresponding DBMS table for each of mutually dependent tables from the relational tables using forward references or ALTER TABLE commands.

54. (original) The system of Claim 50 wherein the operation of creating a corresponding DBMS table comprises:

creating DBMS columns corresponding to columns of the corresponding relational table;

setting properties including precision, scale, length, data type, IsNullable, CollationName, and CharactersetName for each of the DBMS columns based on respective properties of the corresponding relational column;

if one of the DBMS columns is the only one of the DBMS columns that represents a primary key or a foreign key, adding property of primary key or foreign key to the one DBMS column; and

if there is a checkconstraint associated with one of the DBMS columns and not involving any of the remaining DBMS columns, specifying the checkconstraint as column-level constraint.

55. (original) The system of Claim 54 wherein the operation of creating a corresponding DBMS table further comprises:

if there is a multi-column primary key or a multi-column foreign key in the relational table, specifying the multi-column primary key or a multi-column foreign key in the DBMS table at table-level and identifying the DBMS columns that represent the multi-column primary key or a multi-column foreign key; and

if there is a checkconstraint involving multiple DBMS columns, specifying the constraint in the DBMS table at table-level and identifying the involved DBMS columns.

56. (original) The system of Claim 54 wherein the operation of creating a corresponding DBMS table further comprises:

specifying a foreign key in the DBMS table, including:

identifying a child DBMS table and DBMS columns being imported from the child DBMS table; and

specifying properties of the foreign key, the properties including "update" and "delete" referential integrity rules and deferability type.

57. (original) The system of Claim 44 wherein the operation of (7) processing relational views for the first relational schema comprises:

determining relational views associated with the first relational schema;

for each of the relational views:

creating a corresponding DBMS view;

specifying updatability of the corresponding DBMS view; and

specifying query expression defining the corresponding DBMS view.

58. (original) The system of Claim 44 wherein the operation of (8) processing relational indices for the first relational schema comprises:

determining relational indices associated with a first of the relational schemas;

for each of the relational indices:

creating a corresponding DBMS index to represent the relational index;

specifying DBMS columns used by the corresponding DBMS index; and

setting properties of the specified DBMS columns including IsNullable, FilterCondition, and AutoUpdate.

59. (original) The system of Claim 44 wherein the operation of (9) processing relational triggers for the first relational schema comprises:

determining relational triggers associated with the first relational schema;

for each of the relational triggers:

creating a corresponding DBMS trigger;

setting properties of the corresponding DBMS trigger based on properties of the relational trigger, the relational trigger monitoring a relational table; and

setting a monitored DBMS table corresponding to the monitored relational table.

60. (original) The system of Claim 44 wherein the operation of (10) processing relational procedures for the first relational schema comprises:

determining relational procedures associated with the first relational schema;

for each of the relational procedures:

creating a corresponding DBMS procedure; and

setting arguments for the corresponding DBMS procedure based on arguments of the relational procedure.

XI. EVIDENCE APPENDIX

None

XII. RELATED PROCEEDINGS APPENDIX

None